Embedded Software and Hardware for DL

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Course organisation

Sessions

1. Deep Learning and Transfer Learning,
2. Quantization,
3. Pruning,
4. Factorization,
5. Fact. pt.2 : Operators and Architectures,
6. Distillation,
7. Embedded Software and Hardware for DL,
8. Presentations for challenge.

What are the potential targets ?

- CPU
- GPU
- ASICs
  - IPU (Graphcore)
  - TPU (Google)
  - Edge TPU (Google)
  - Eyeriss (MIT)
  - ...
- FPGA

What are the potential targets ?

- CPU
- GPU
- ASICs
  - IPU (Graphcore)
  - TPU (Google)
  - Edge TPU (Google)
  - Eyeriss (MIT)
  - ...
- FPGA

Questions

- What are the differences between them ?
- Which use case for each target ?

What are the potential targets ?

- CPU
- GPU
- ASICs
  - IPU (Graphcore)
  - TPU (Google)
  - Edge TPU (Google)
  - Eyeriss (MIT)
  - ...
- FPGA

Questions

- What are the differences between them ?
- Which use case for each target ?

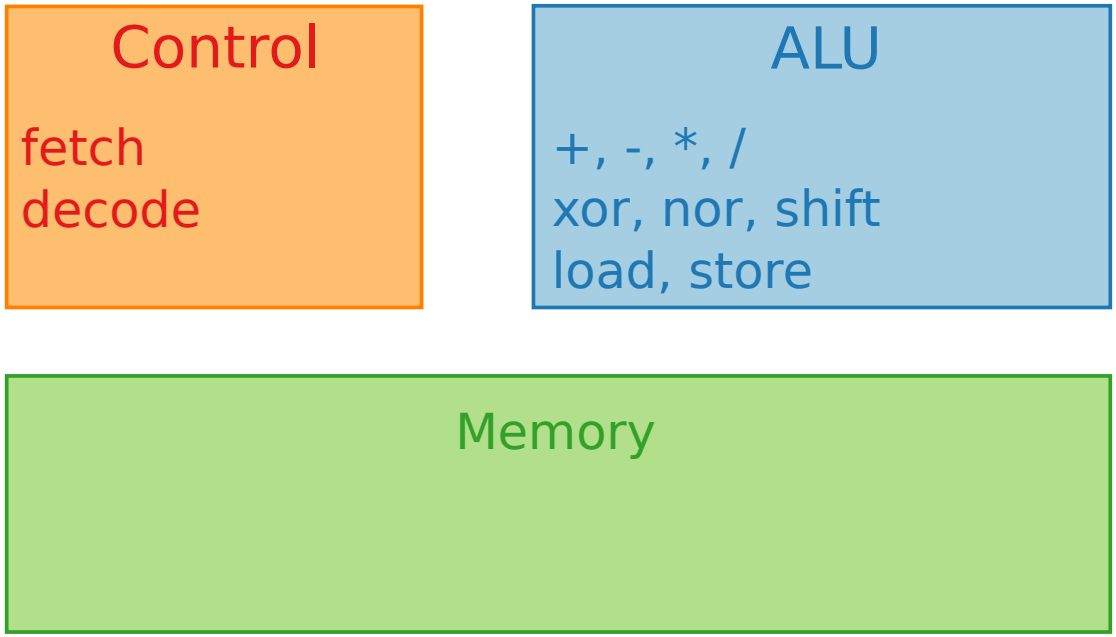
What are the potential targets ?

- CPU: What are the elements of a CPU ?
- GPU
- ASICs
  - IPU (Graphcore)
  - TPU (Google)
  - Edge TPU (Google)
  - Eyeriss (MIT)
  - ...
- FPGA

Questions

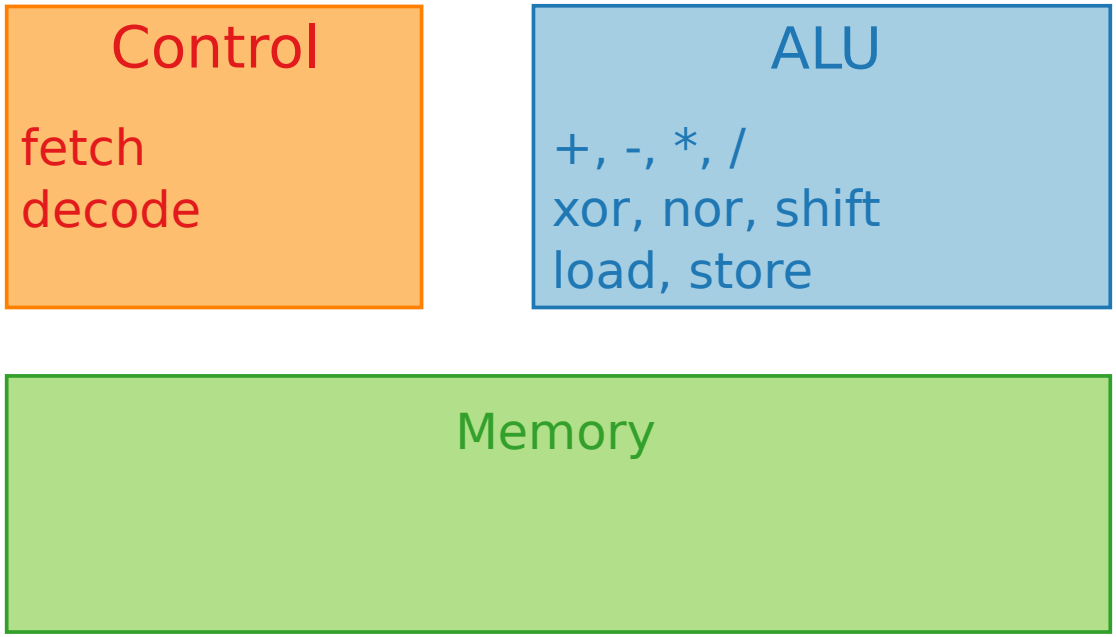
- What are the differences between them ?
- Which use case for each target ?

What are the elements of a CPU ?



- Control: Fetches and decodes instructions, controls the ALU,
- ALU: Arithmetical and Logical Unit, performs all computations, exchanges data between memory and register file,
- Memory: Stores data.

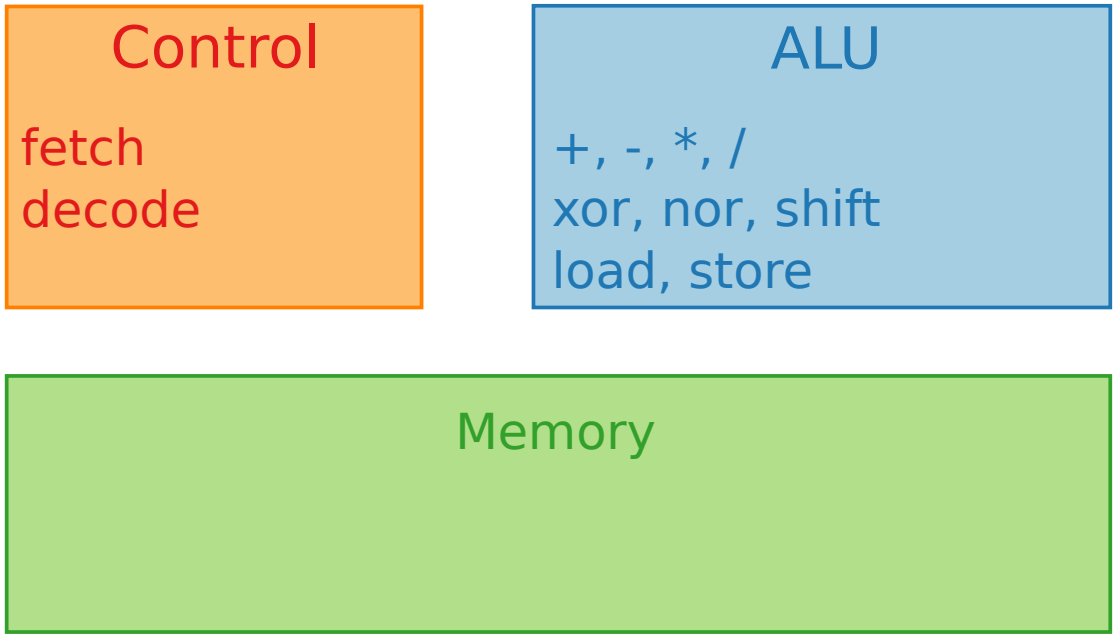
What are the elements of a CPU ?



There are many ways to increase the overall performance of a CPU architecture. The reader may refer to the following book for a broad study of the field.

J. L. Hennessy and D. A. Patterson, *Computer Architecture, Sixth Edition: A Quantitative Approach*, 6th. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2017, ISBN: 0128119055.

What are the elements of a CPU ?



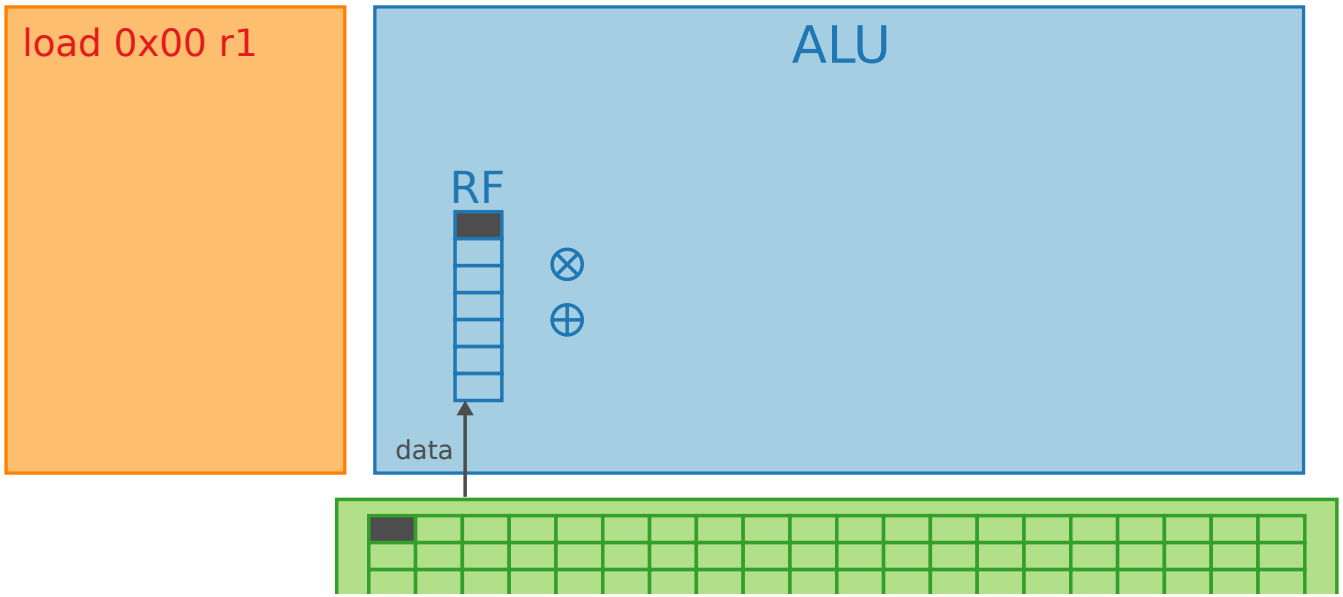
In this course, two key features will be described:

- Increasing the computational parallelism,
- Reducing data accesses time with close and fast memories.

Increasing Parallelism : SIMD

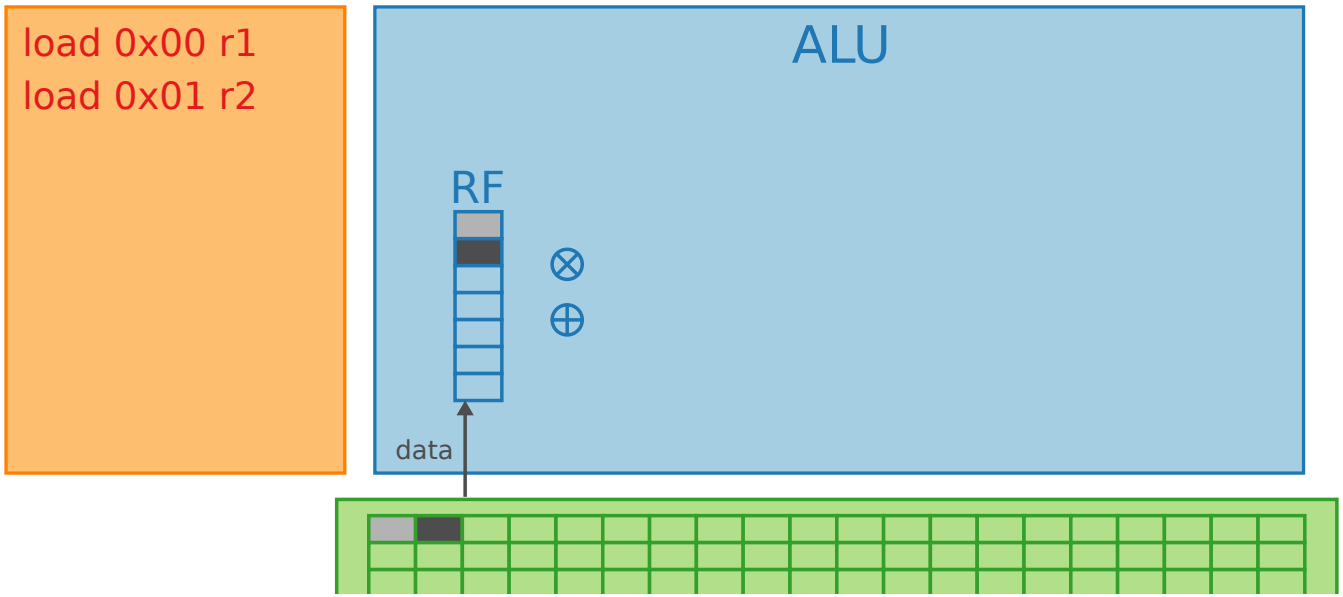
- SIMD: Single Instruction Multiple Data
- Hardware feature in ALU
- Available in Intel CPUs (SSE, AVX)
- Available in ARM CPUs (Neon)

Increasing Parallelism : SIMD



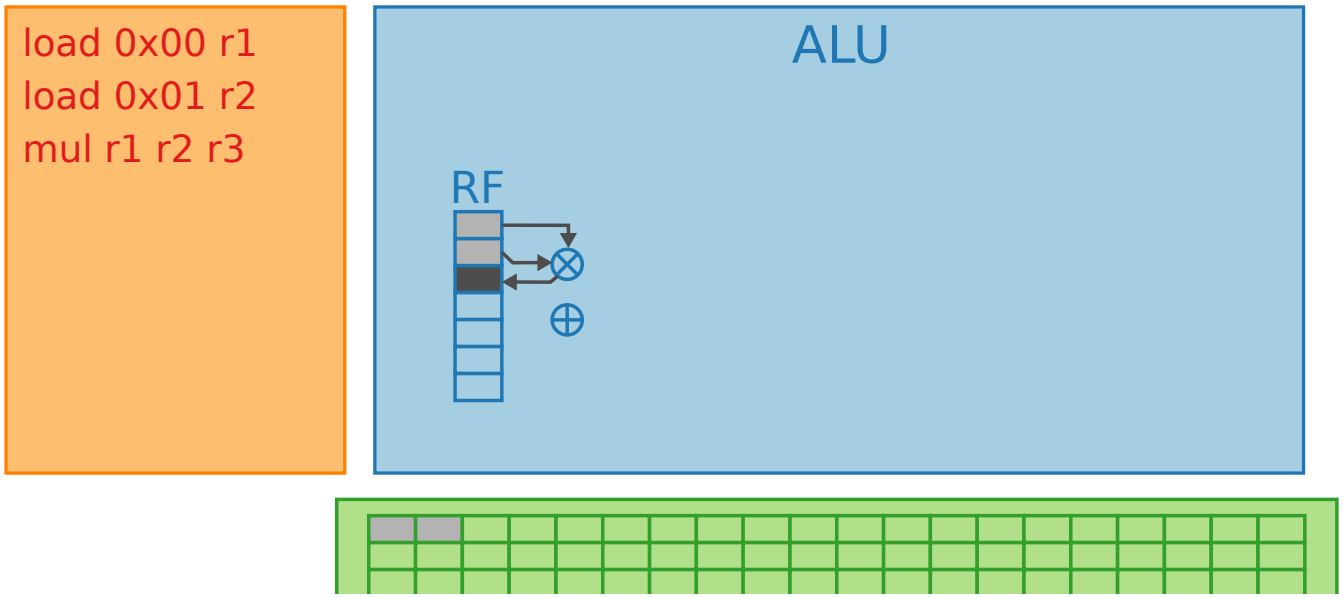
- "Normal" Single Instruction Single Data (SISD) example
- Load data from memory to register file
- Execute multiplication
- Execute addition

Increasing Parallelism : SIMD



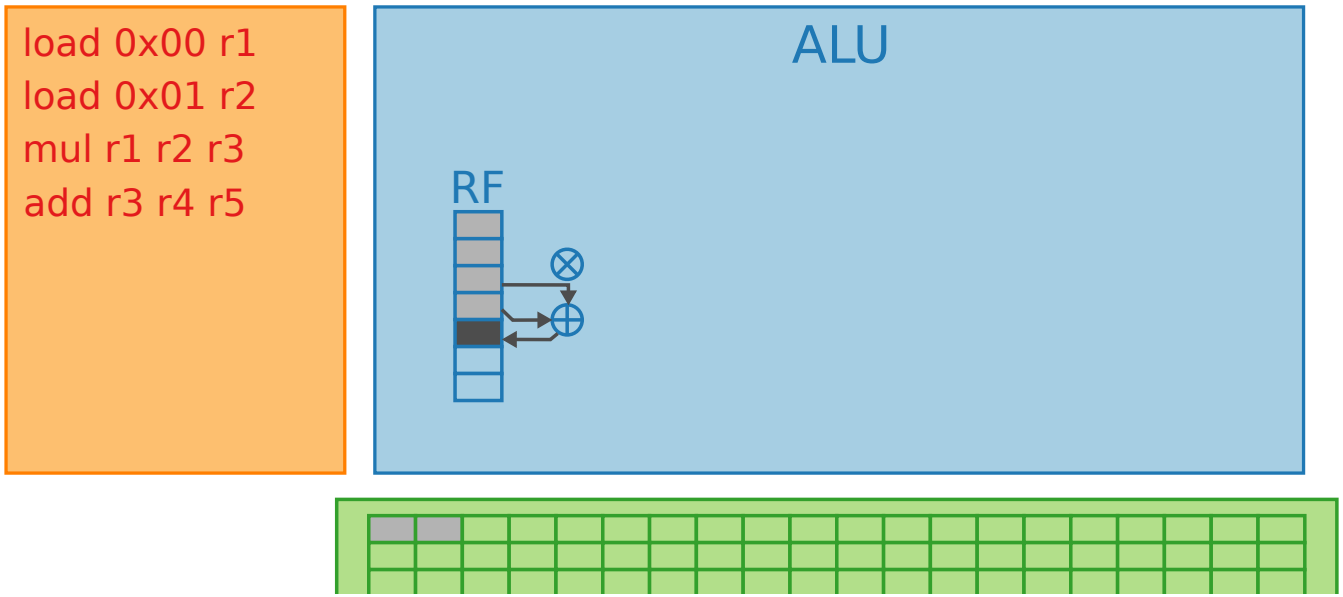
- "Normal" Single Instruction Single Data (SISD) example
- Load data from memory to register file
- Execute multiplication
- Execute addition

Increasing Parallelism : SIMD



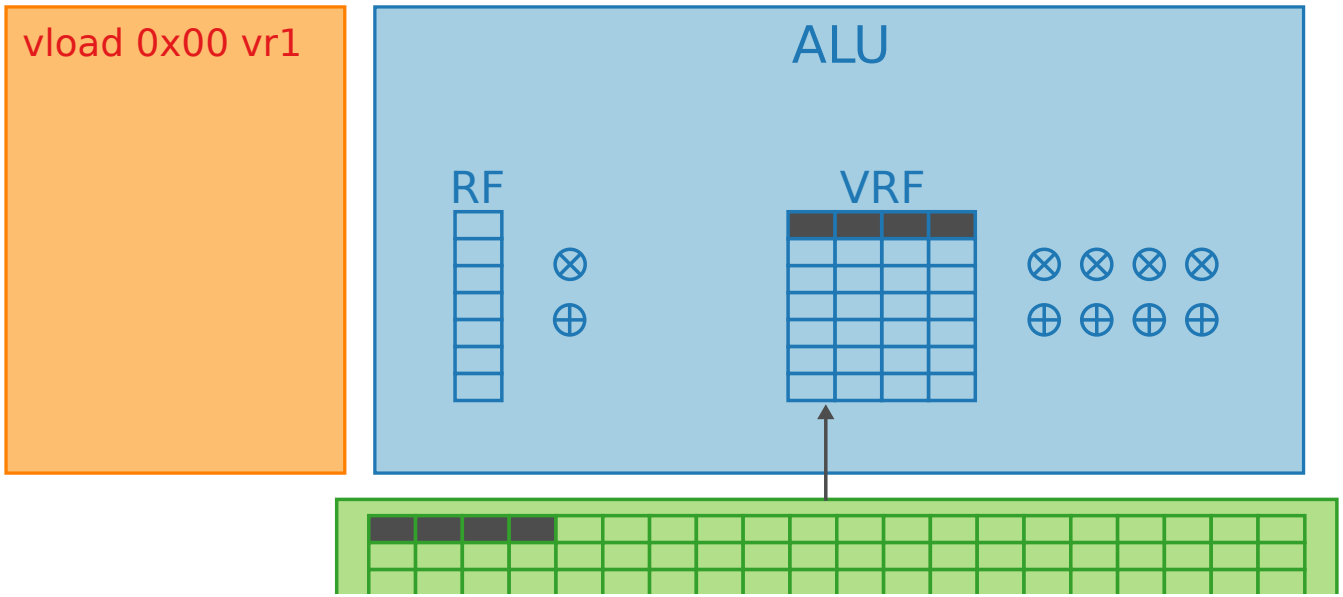
- "Normal" Single Instruction Single Data (SISD) example
- Load data from memory to register file
- Execute multiplication
- Execute addition

Increasing Parallelism : SIMD



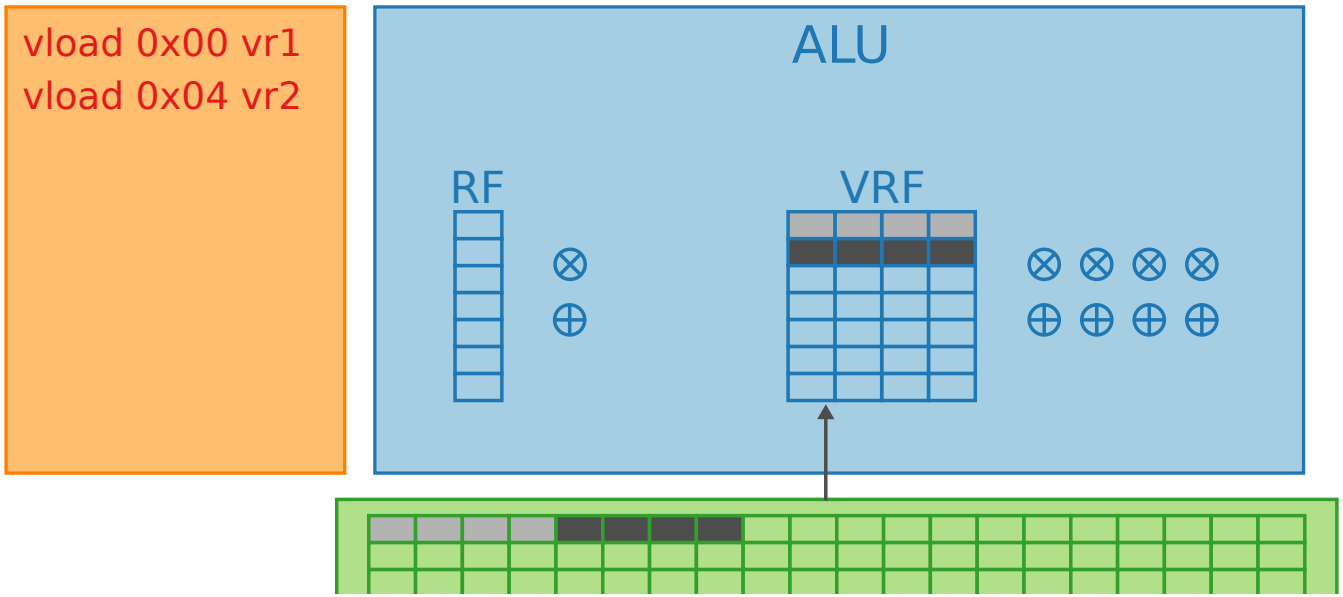
- "Normal" Single Instruction Single Data (SISD) example
- Load data from memory to register file
- Execute multiplication
- Execute addition

Increasing Parallelism : SIMD



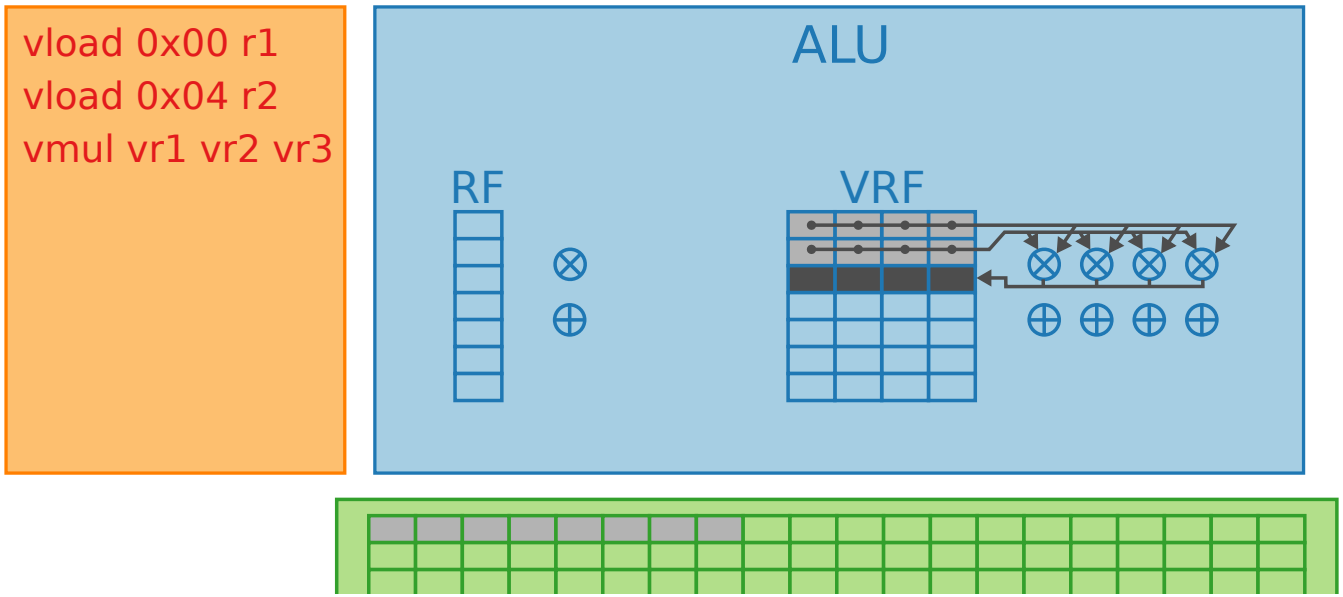
- Single Instruction Multiple Data
- Additional hardware
- Parallel load
- Parallel arithmetic
- Increase number of computations per instruction

Increasing Parallelism : SIMD



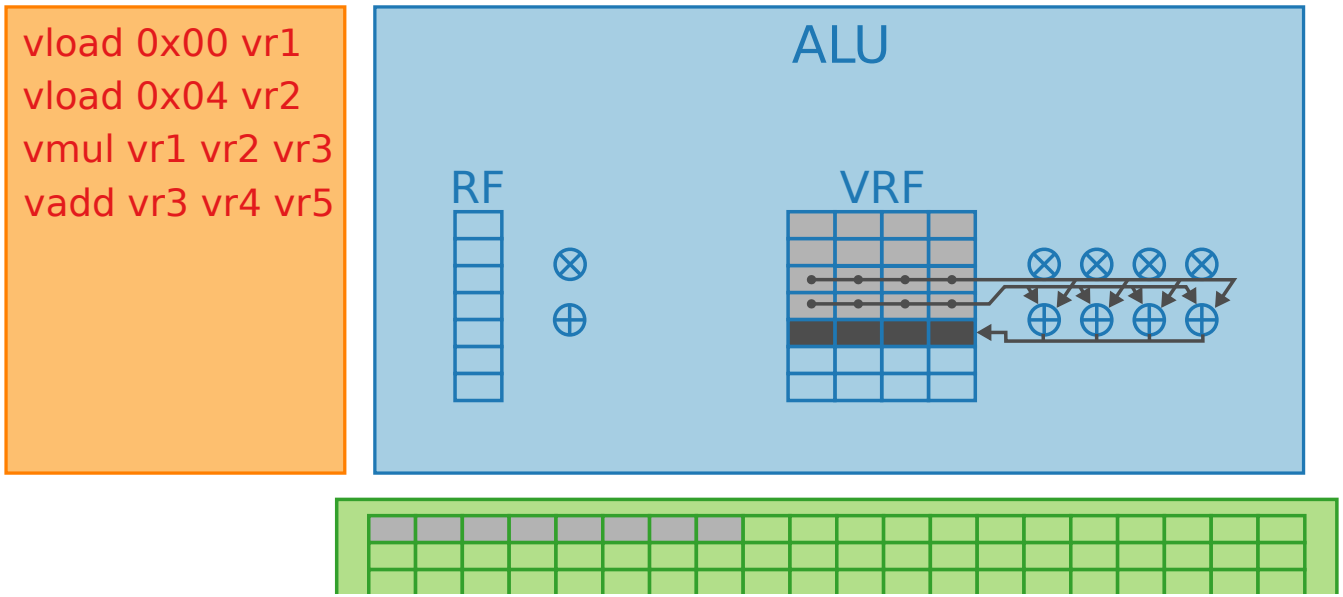
- Single Instruction Multiple Data
- Additional hardware
- Parallel load
- Parallel arithmetic
- Increase number of computations per instruction

Increasing Parallelism : SIMD



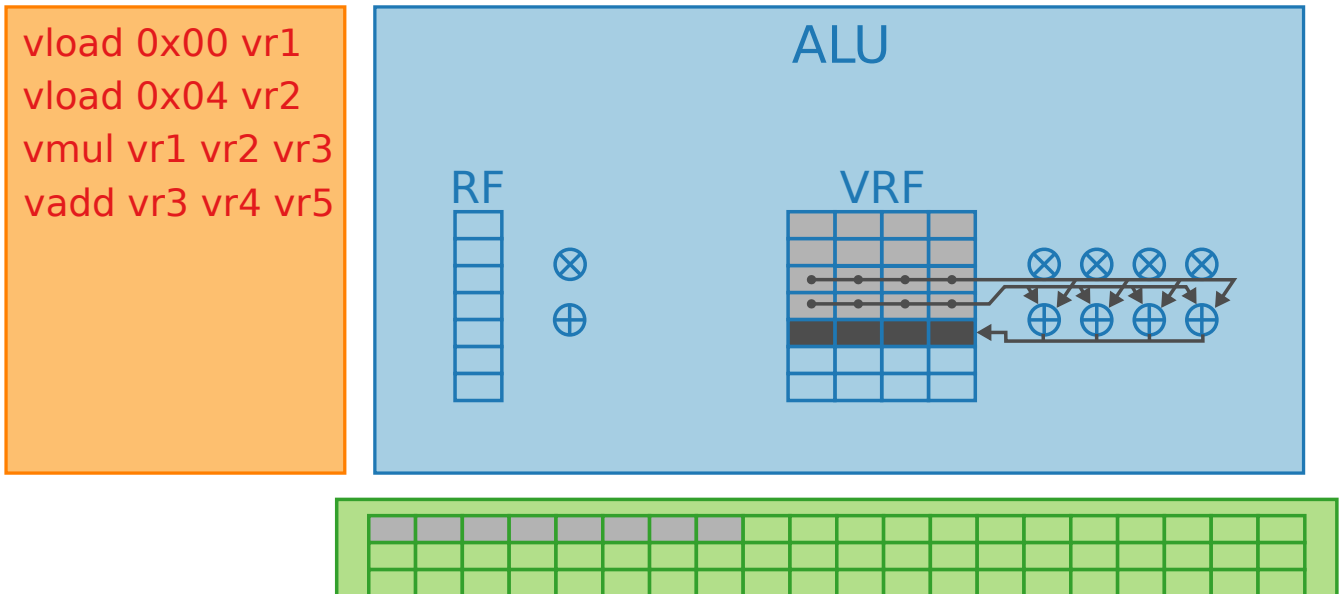
- Single Instruction Multiple Data
- Additional hardware
- Parallel load
- Parallel arithmetic
- Increase number of computations per instruction

Increasing Parallelism : SIMD



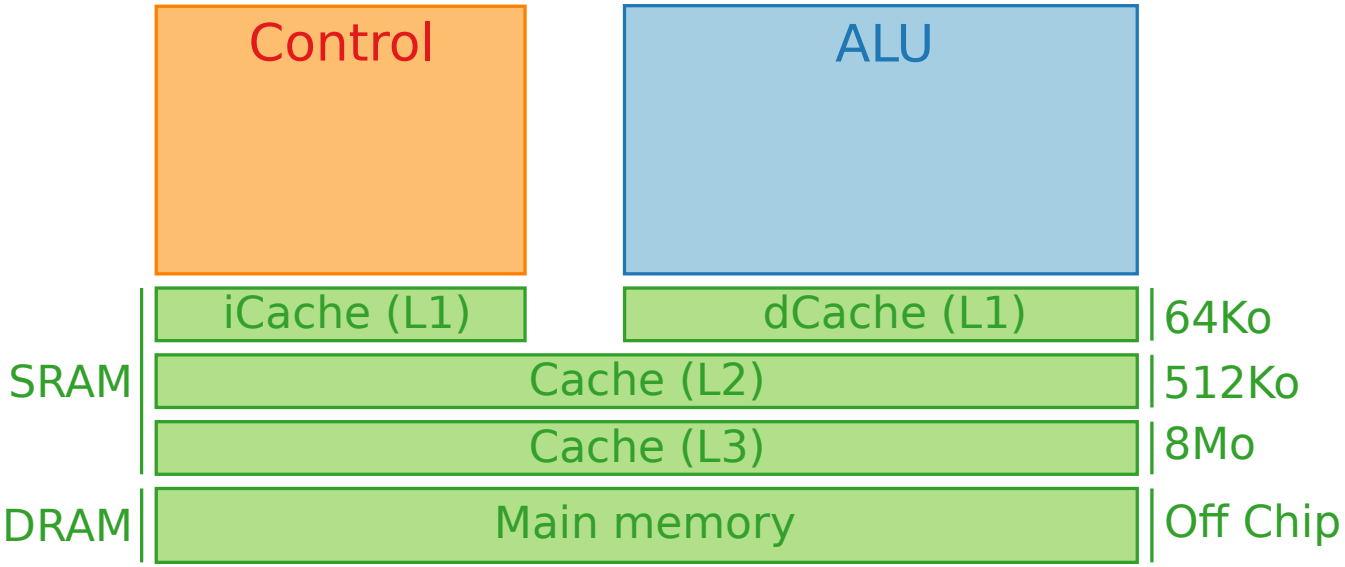
- Single Instruction Multiple Data
- Additional hardware
- Parallel load
- Parallel arithmetic
- Increase number of computations per instruction

Increasing Parallelism : SIMD



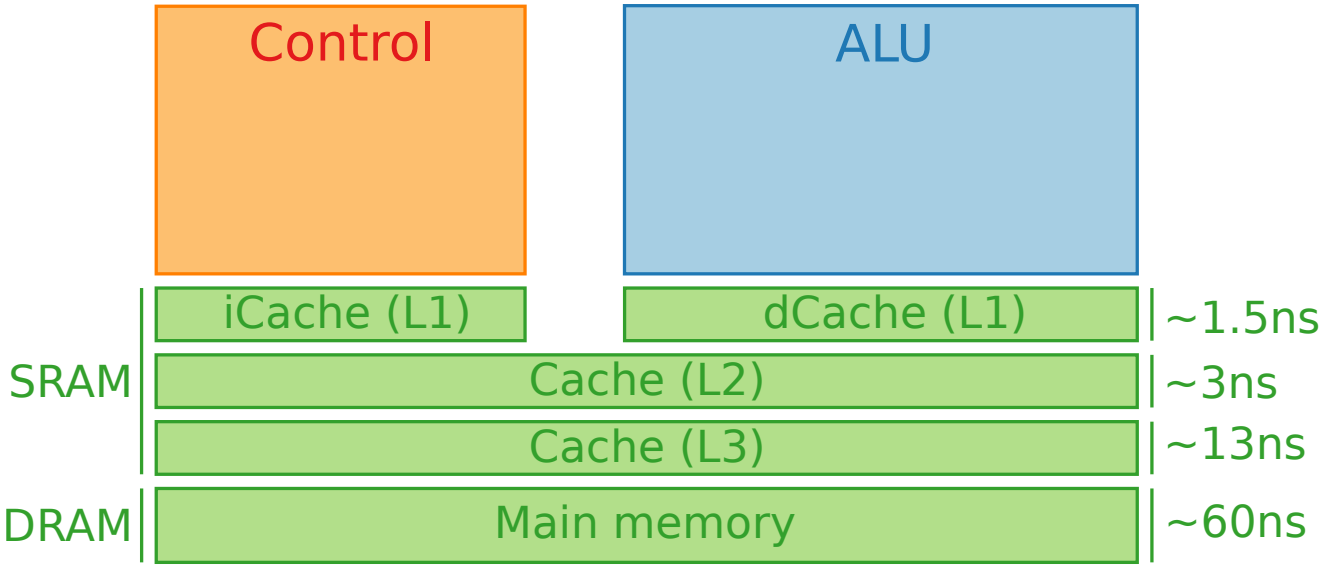
- Increased parallelism
- Multiple quantization formats handled (8-, 16-, 32-, 64-bit)
- The more quantized, the more parallel
- Need aligned data in memory

Cache hierarchy



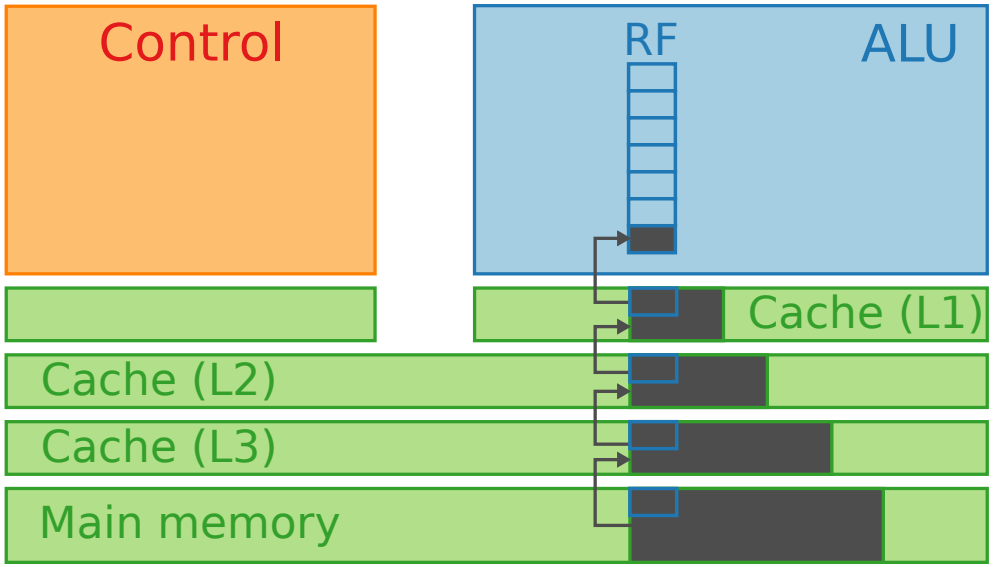
- Cache Hierarchy
- SRAM vs DRAM
- First Access
- Cache Hit
- Cache Miss

Cache hierarchy



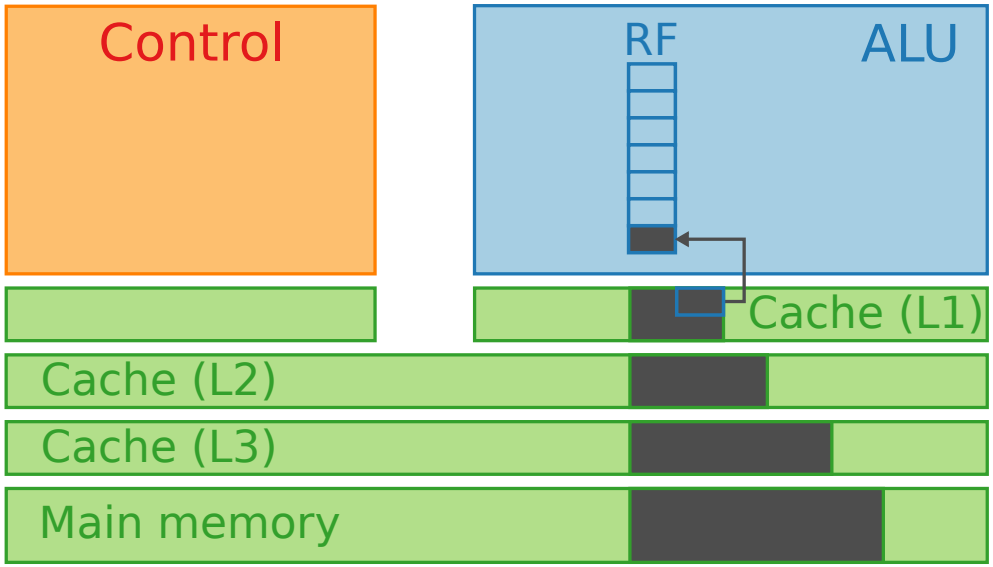
- Cache Hierarchy
- SRAM vs DRAM
- First Access
- Cache Hit
- Cache Miss

Cache hierarchy



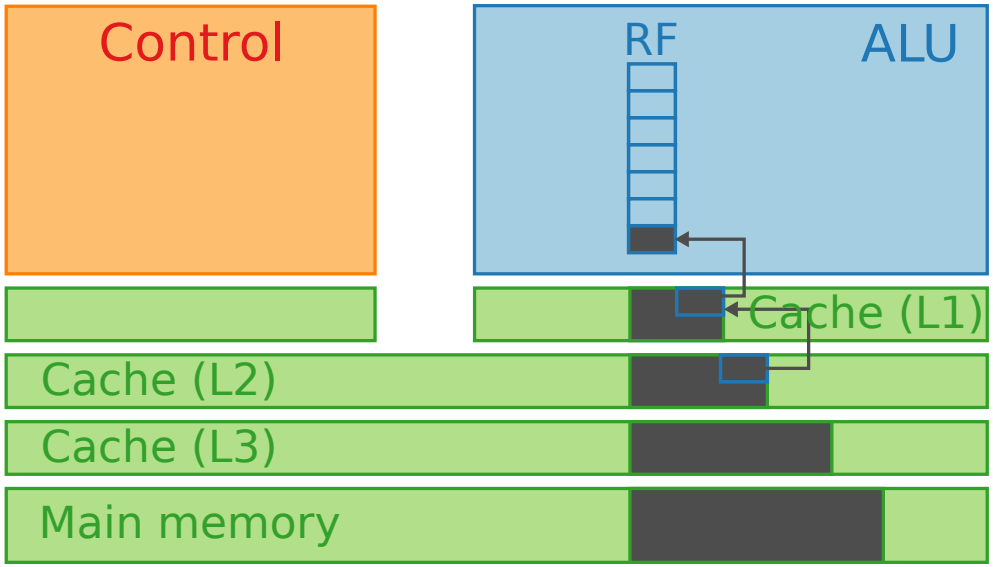
- Cache Hierarchy
- SRAM vs DRAM
- First Access
- Cache Hit
- Cache Miss

Cache hierarchy



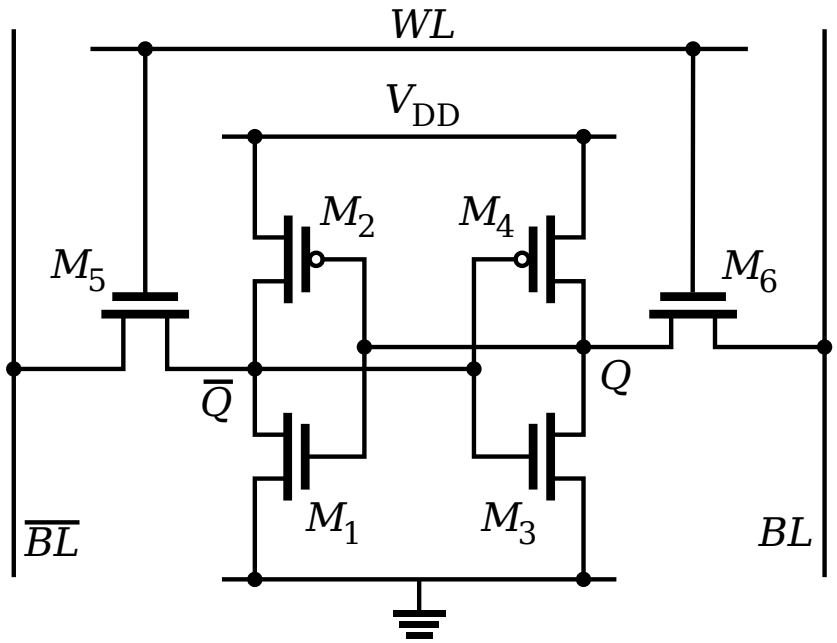
- Cache Hierarchy
- SRAM vs DRAM
- First Access
- Cache Hit
- Cache Miss

Cache hierarchy

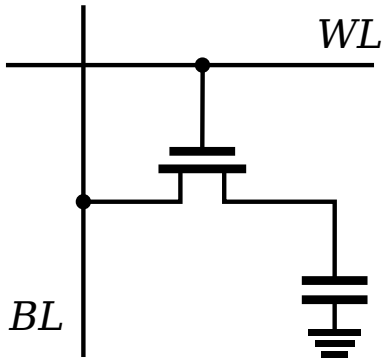


- Cache Hierarchy
- SRAM vs DRAM
- First Access
- Cache Hit
- Cache Miss

SRAM vs DRAM



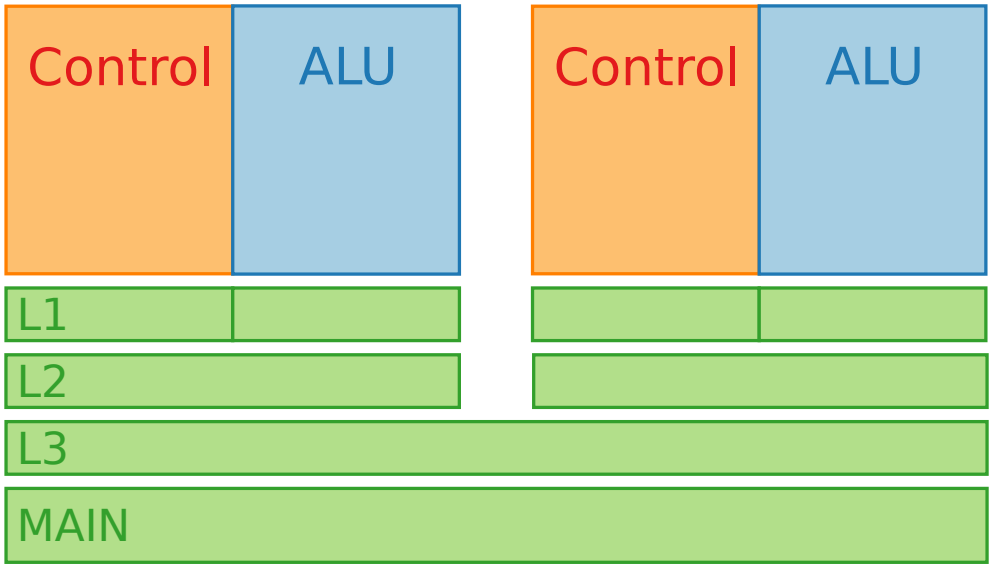
SRAM



DRAM

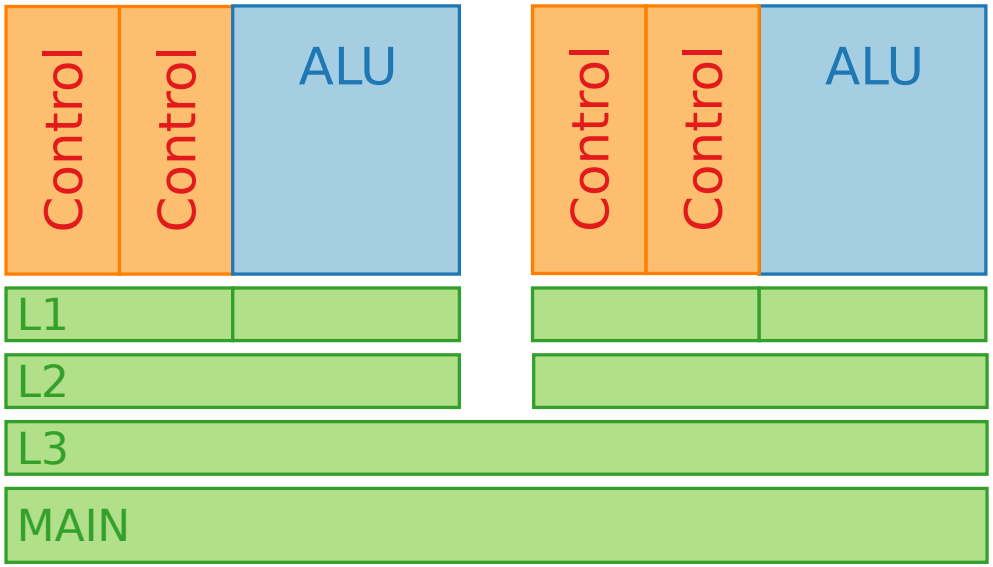
- SRAM 6T (typically) vs DRAM 1T
- SRAM is more expensive
- DRAM is denser
- DRAM needs refreshment
- SRAM is faster

Multicore



- Add CPU cores on the same chip
- Last Level Cache (LLC) is shared between cores
- Linear increasing of computing capacity

Simultaneous Multi Threading (SMT)

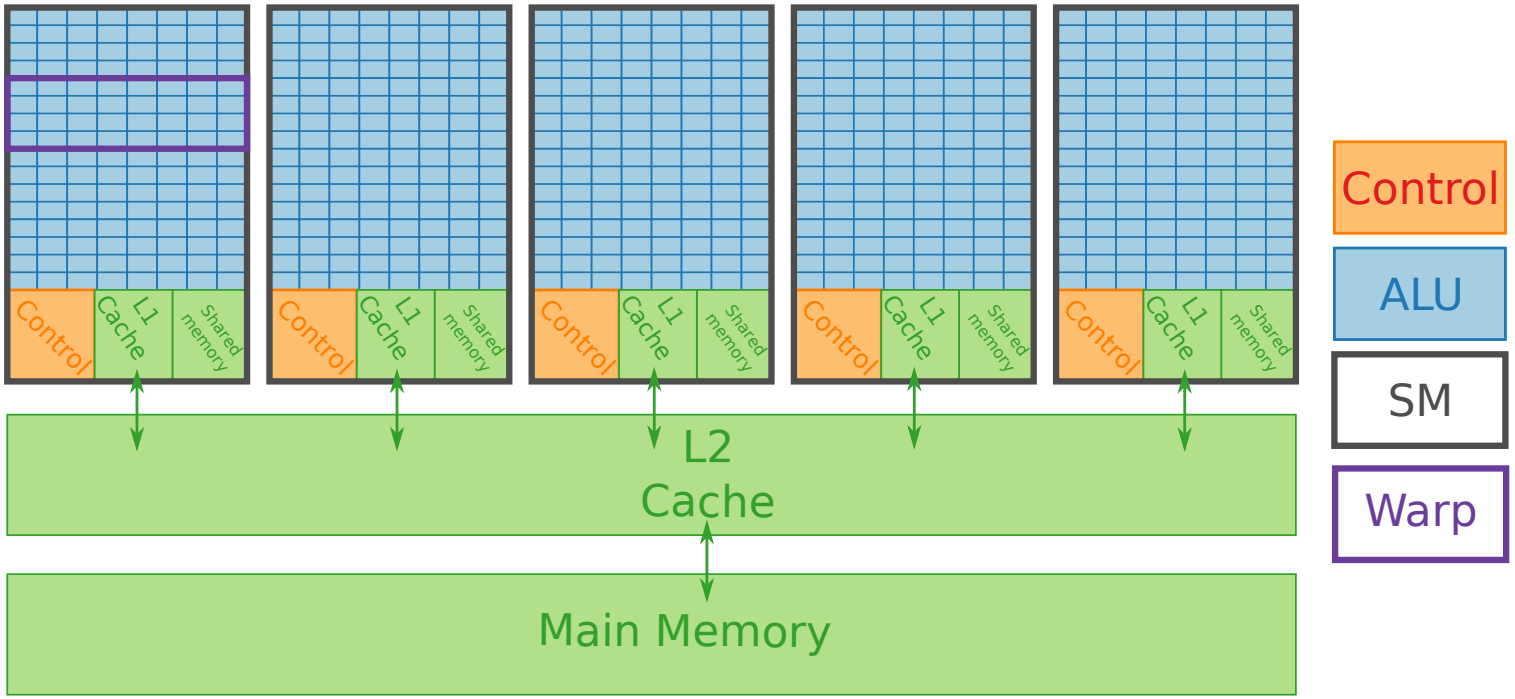


- Known as "Hyperthreading" which is Intel's own SMT implementation
- Multiple instruction threads (here 2) are processed on each core
- Sublinear increasing of computing capacity, resources are shared

GPU

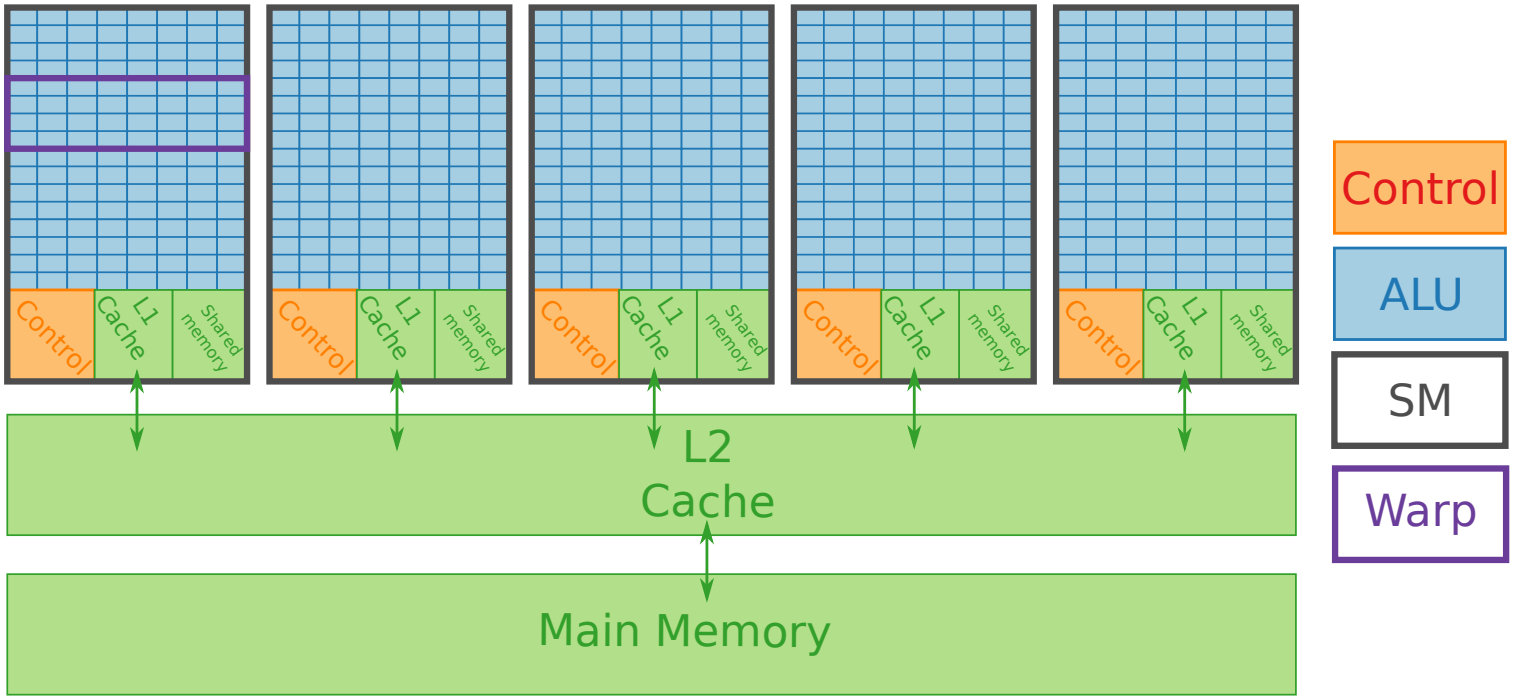
- CPU
- GPU
- ASICs
  - IPU (Graphcore)
  - TPU (Google)
  - Edge TPU (Google)
  - Eyeriss (MIT)
  - ...
- FPGA

GPU



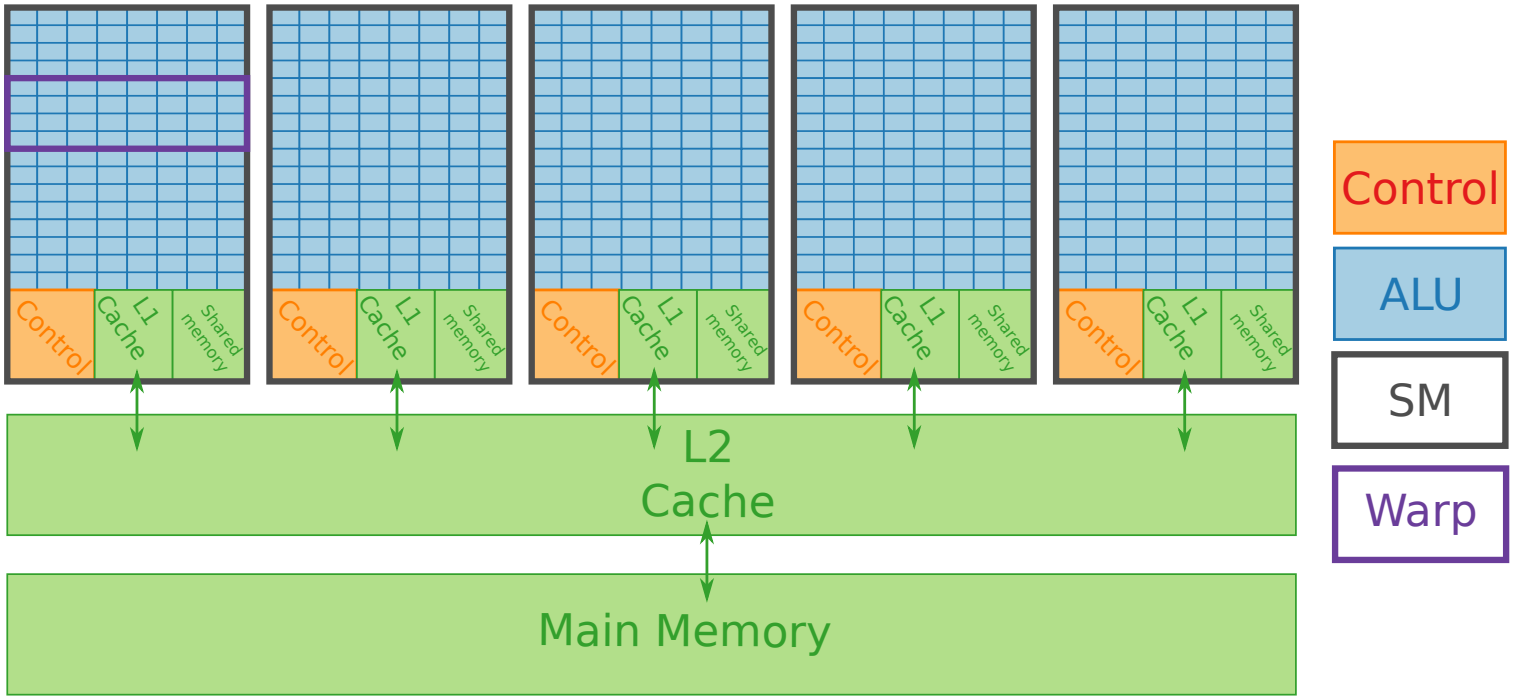
- GPUs have a huge computation power
- Simpler control
- Each core execute warps of 32 threads (Nvidia)
- Same instructions in each thread, but different execution contexts
- Yields higher throughput, but also higher latency

GPU



- GPUs have a huge computation power
- Simpler control
- Each core execute warps of 32 threads (Nvidia)
- Same instructions in each thread, but different execution contexts
- Yields higher throughput, but also higher latency

GPU



- GPUs have a huge computation power
- Simpler control
- Each core execute warps of 32 threads (Nvidia)
- Same instructions in each thread, but different execution contexts
- Yields higher throughput, but also higher latency

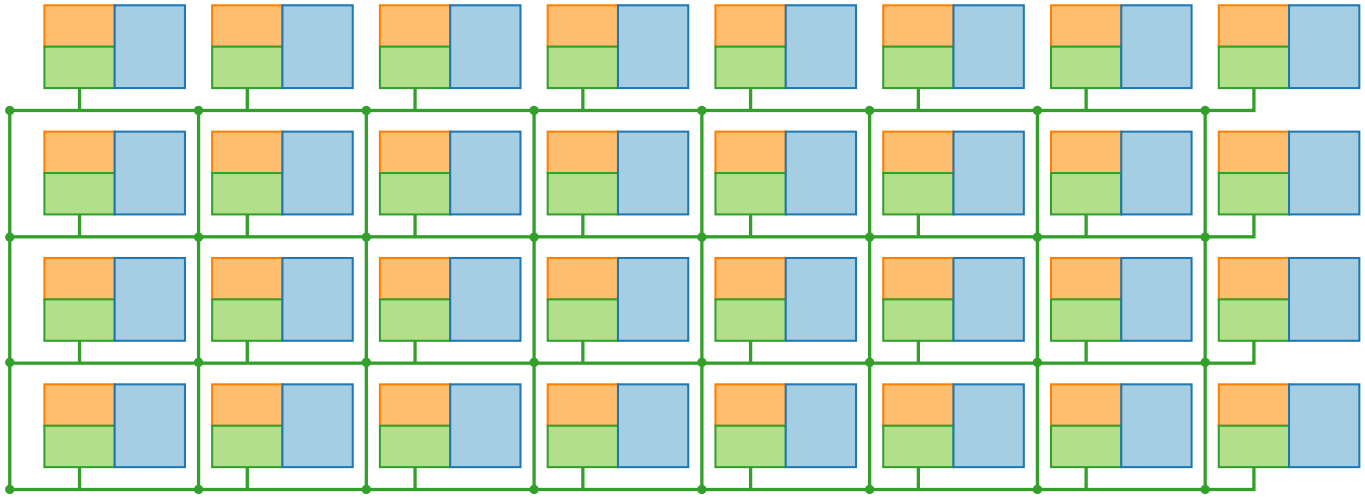
ASICs

- CPU
- GPU
- ASICs : Application Specific Integrated Circuits
  - IPU (Graphcore)
  - TPU (Google)
  - Edge TPU (Google)
  - Eyeriss (MIT)
  - ...
- FPGA

ASICs

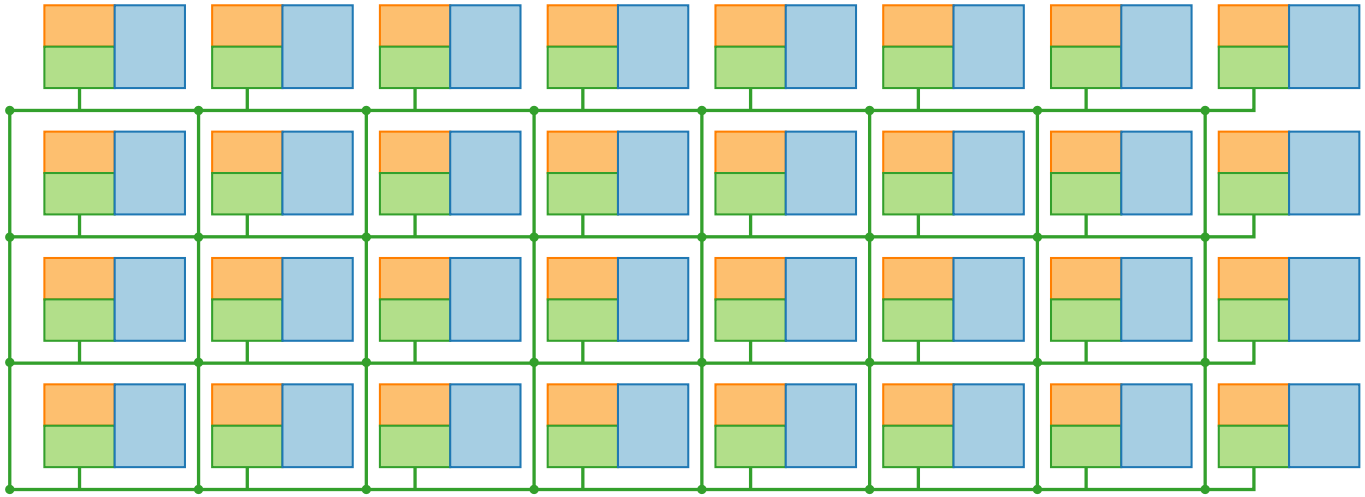
- CPU
- GPU
- ASICs : Application Specific Integrated Circuits
  - IPU (Graphcore)
  - TPU (Google)
  - Edge TPU (Google)
  - Eyeriss (MIT)
  - ...
- FPGA

ASICs : Example of Graphcore's IPU



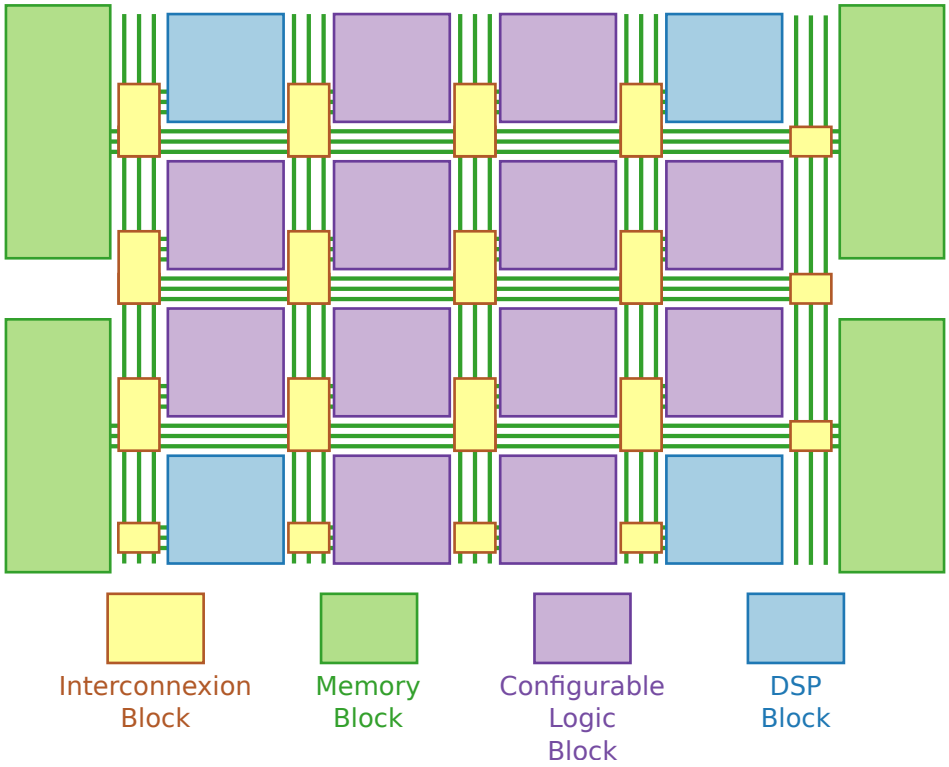
- Manycore approach :
- Each core handles 6 independent threads
- Fully distributed cache memory
- 256Ko / core

ASICs : Example of Graphcore's IPU



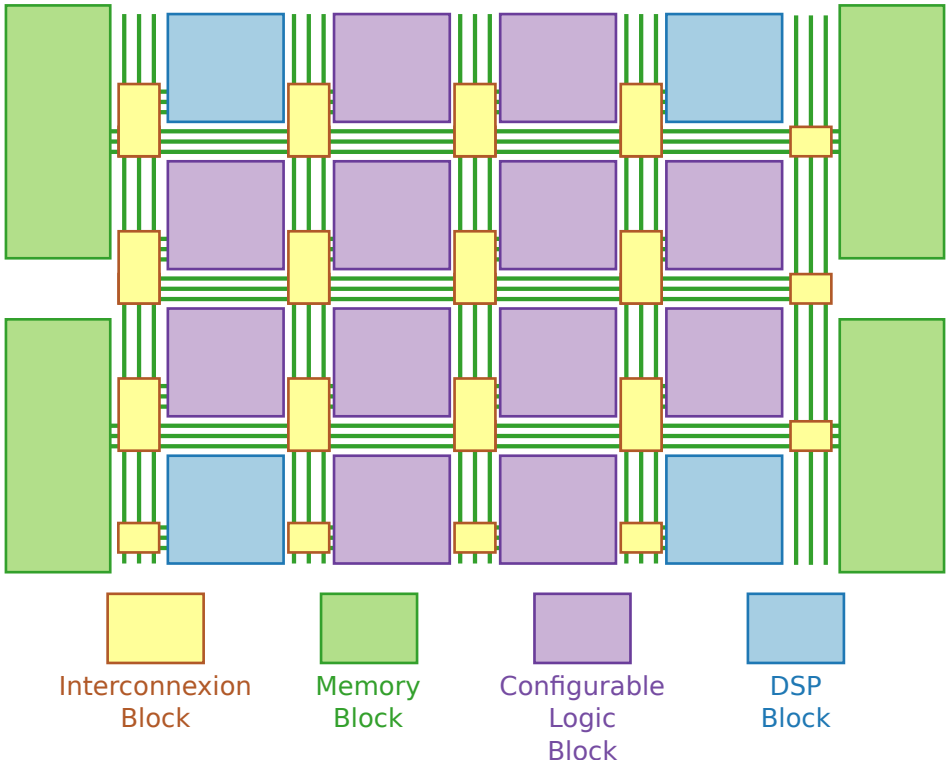
- Claims better efficiency ($/Gops, kWh/Gops)
- Claims faster inference
- Cautious: lack of independent benchmarks

FPGAs : (Re)Configurable Integrated Circuits



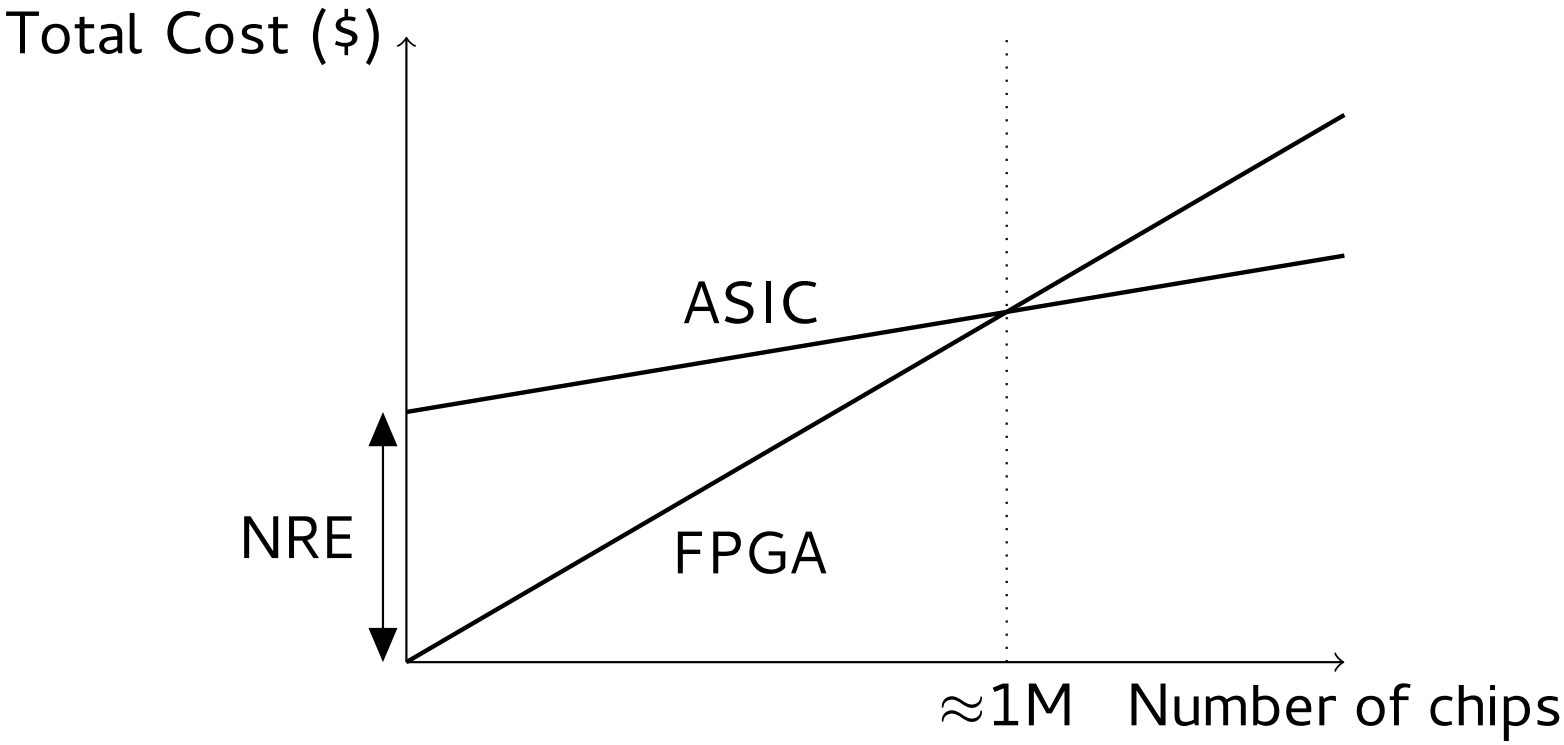
- Designing a custom architecture
- No "Non Recurring Engineering" compared to custom ASIC
- Prototyping
- Small markets

FPGAs : (Re)Configurable Integrated Circuits



- Designing a custom architecture
- No "Non Recurring Engineering" compared to custom ASIC
- Prototyping
- Small markets

FPGAs : (Re)Configurable Integrated Circuits



- Designing a custom architecture
- No "Non Recurring Engineering" compared to custom ASIC
- Prototyping
- Small markets

Remote vs Local use cases

Use case
Remote

Remote vs Local use cases

Use case

Remote

Key features

- Throughput
- Cost ($/Gops)
- Scaling

Remote vs Local use cases

Use case
Remote

Key features

- Throughput
- Cost ($/Gops)
- Scaling

Targets

- GPU
- TPU
- IPU

Remote vs Local use cases

Use case

Remote

Key features

- Throughput
- Cost ($/Gops)
- Scaling

Targets

- GPU
- TPU
- IPU

Use case

Local

Remote vs Local use cases

Use case

Remote

Key features

- Throughput
- Cost ($/Gops)
- Scaling

Targets

- GPU
- TPU
- IPU

Use case

Local

Key features

- Availability
- Power consumption
- Cost ($/unit)
- Latency
- Data privacy

Remote vs Local use cases

Use case

Remote

Key features

- Throughput
- Cost ($/Gops)
- Scaling

Targets

- GPU
- TPU
- IPU

Use case

Local

Key features

- Availability
- Power consumption
- Cost ($/unit)
- Latency
- Data privacy

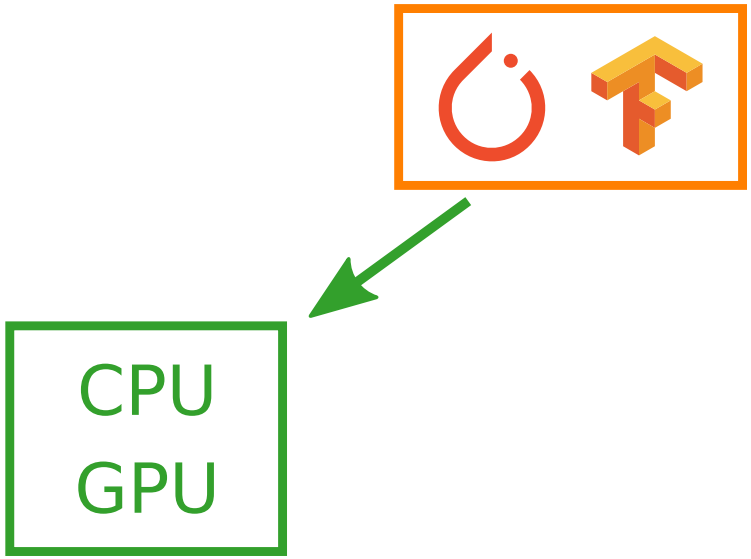
Targets

- CPU
- Edge TPU
- Embedded GPU (Tegra)
- FPGA

And what about software ?

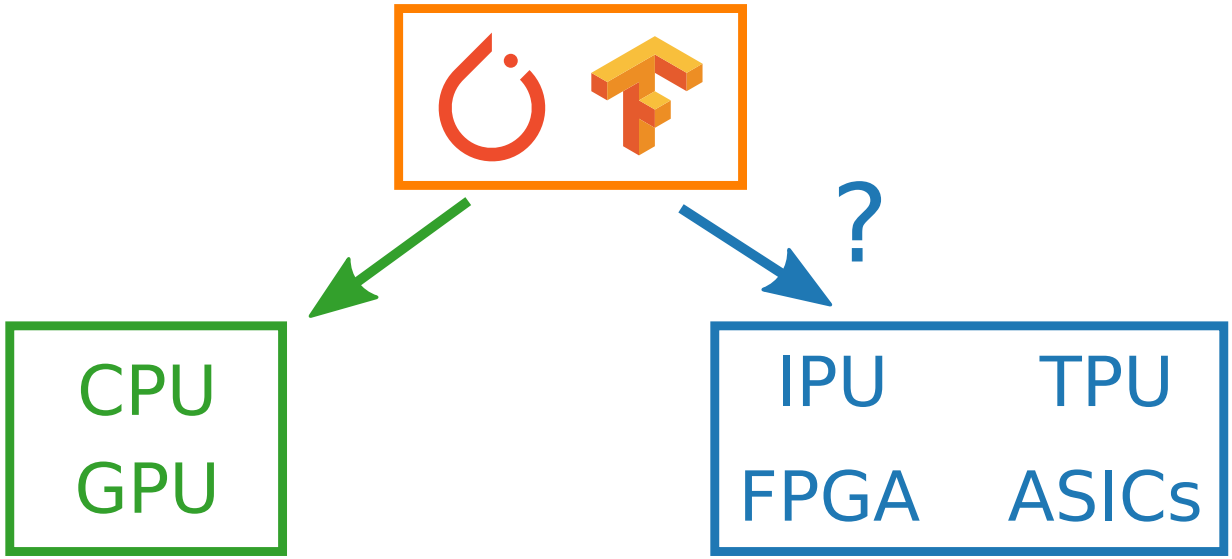
- High level frameworks
- Broadly used
- Programmed and optimized to be used on CPU and GPU
- Not systematically ported on each target
- Supporting these frameworks becomes critical for chips makers

And what about software ?



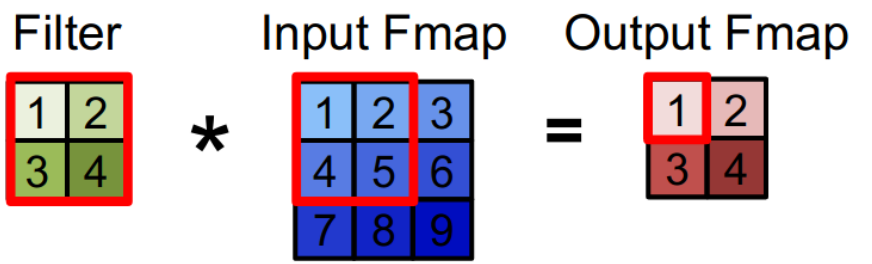
- High level frameworks
- Broadly used
- Programmed and optimized to be used on CPU and GPU
- Not systematically ported on each target
- Supporting these frameworks becomes critical for chips makers

And what about software ?

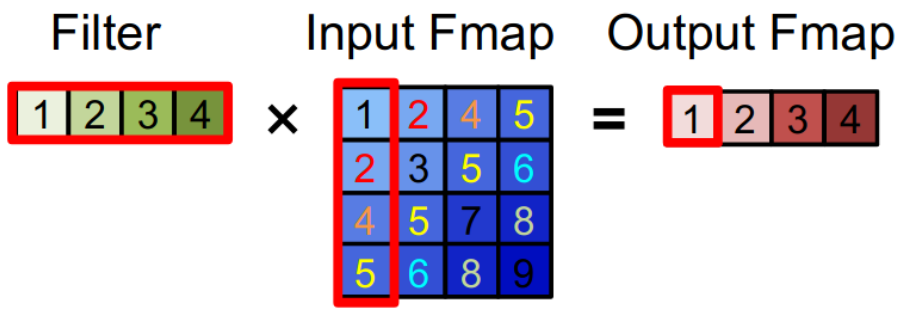


- High level frameworks
- Broadly used
- Programmed and optimized to be used on CPU and GPU
- Not systematically ported on each target
- Supporting these frameworks becomes critical for chips makers

Software: matrix multiplication



Convolution



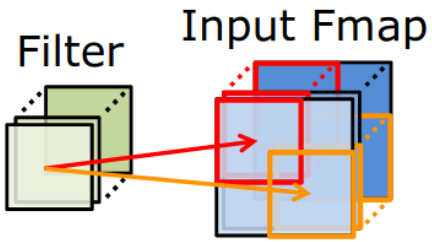
Matrix Multiply (by Toeplitz Matrix)

Data is repeated

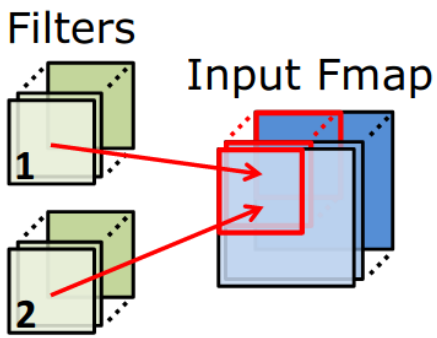
- Use existing optimized libraries
- Repeating Data

From: `http://eyeriss.mit.edu/2019_neurips_tutorial.pdf`

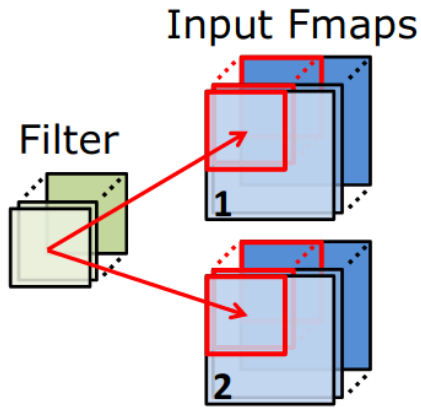
Software: data reuse



Convolutional Reuse
(Activations, Weights)
CONV layers only
(sliding window)



Fmap Reuse
(Activations)
CONV and FC layers



Filter Reuse
(Weights)
CONV and FC layers
(batch size > 1)

48

- Keep data in caches
- Activations and / or weights

From: `http://eyeriss.mit.edu/2019_neurips_tutorial.pdf`